3. Учебные вопросы:

1. Создание таблиц и схем данных

2. Описание выражений для индексирования ключей, используемых для организации связей между таблицами данных.

Вопрос 1. Создание таблиц и схем данных

Как уже отмечалось ранее, процесс разработки базы данных начинается с задания описания структур таблиц. Рассмотрим этот процесс более подробно. Итак, для начала нам необходимо создать описание таблицы Бумаги. Нажав кнопку Создать и выбрав в появившемся вслед диалоговом окне режим Конструктор, мы попадаем в окно, предназначенное для ввода описания структуры создаваемой таблицы. Оно изображено на рис. 1. При создании баз данных, предназначенных для решения финансовых и экономических задач, процесс описания атрибутов полей в создаваемой таблице приобретает особое значение. Как видно из рис. 1, процесс описания атрибутов поля начинается с присвоения ему имени (идентификатора). Желательно, чтобы это имя было, с одной стороны, информативным, а с другой - кратким, что обеспечивает несомненные удобства при дальнейших манипуляциях с ним. Далее необходимо определить тип поля, что, очевидно, должно делаться, исходя из содержания тех данных, которые будут в нем храниться.

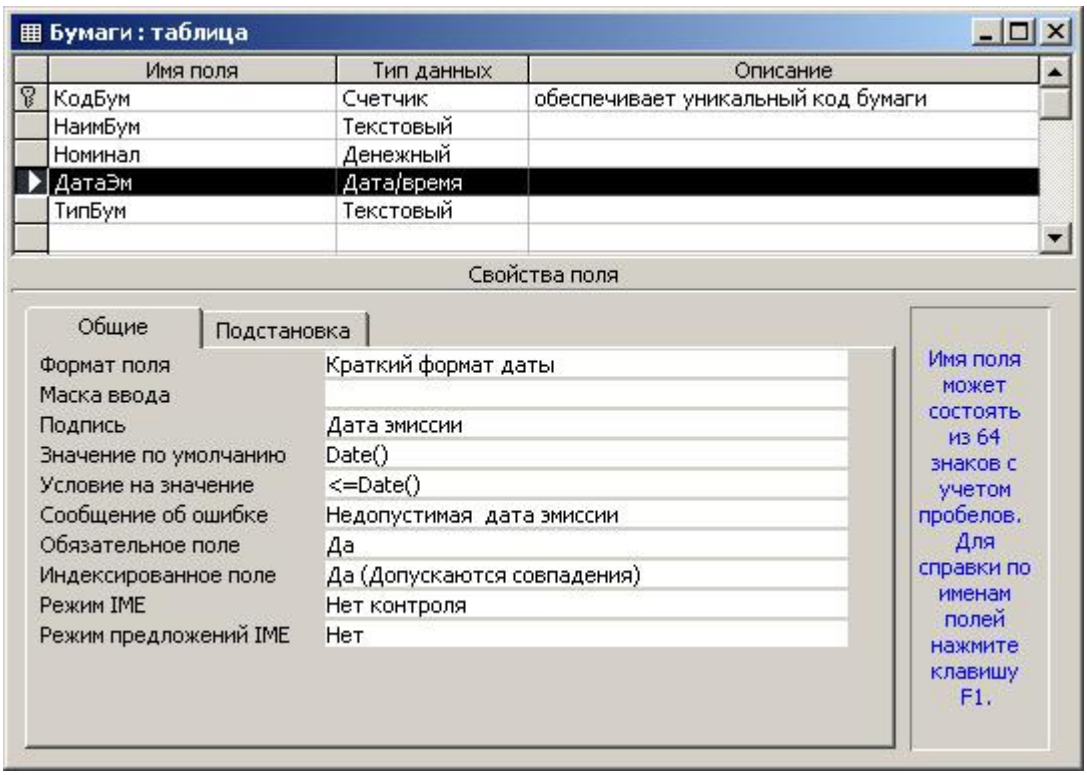

Рис. 1 Создание описания структуры таблицы Бумаги

Обратим внимание на тип Счетчик, присвоенный полю КодБум (код бумаги). Он означает, что СУБД будет самостоятельно помещать в это поле

некоторое числовое значение для каждой вновь создаваемой записи таблицы, обеспечивая таким образом его уникальность.
Выбор типа данных в Access одновременно определяет набор дополнительных атрибутов соответствующего поля. В частности, поле ДатаЭм (дата эмиссии) имеет тип Дата и, как это показано на рис. 1, может иметь дополнительные атрибуты:
- формат поля, определяющий условия вывода данных из этого поля (по умолчанию);
" Маска ввода, определяющая условия ввода данных в поле;
- подпись - содержит расширенный заголовок;
- значение по умолчанию - позволяет указать значение, автоматически присваиваемое полю при создании новой записи. В нашем случае по умолчанию будет задаваться текущая дата, возвращаемая встроенной функцией Date();
- условие на значение - определяет требования к данным, вводимым в поле. Например, для выполнения требования, чтобы дата эмиссии предшествовала текущей, следует задать выражение <=Date();
- сообщение об ошибке - определяет текст сообщения, которое будет выводиться в случае нарушения заданного выше условия;
- обязательное поле - указывает, требует или нет поле обязательного ввода значения;
- индексированное поле - определяет индекс, создаваемый по данному полю. Индекс ускоряет выполнение запросов, в которых используются индексированные поля, и операции сортировки и группировки.
Основываясь на опыте проектирования различных баз, необходимо заметить, что не следует пренебрегать возможностями управления данными, которые открывают дополнительные атрибуты полей. Их грамотное и продуманное использование позволяет организовать централизованный и эффективный контроль за корректностью и целостностью данных.
На завершающем этапе процесса проектирования структуры таблицы происходит задание ключей и индексов. В первом случае достаточно выделить строки, которые должны составить ключевое выражение, и щелкнуть мышью по пиктограмме Ключ на панели инструментов (рис. 2). В таблице Бумаги роль уникального ключевого идентификатора выполняет поле КодБум.

Рис. 2 Панель инструментов конструктора таблиц

Также при создании таблицы имеет смысл заранее продумать возможные упорядочения, которые могут понадобиться при работе с содержащимися в ней данными. Задание индексов с соответствующими ключевыми выражениями может в дальнейшем существенно ускорить процесс работы (особенно с большими массивами данных). В частности, при работе с данными из таблицы Бумаги весьма вероятно, что нам придется выводить их в алфавитном порядке

по названиям, а также отсортированными в порядке убывания дат. Процесс задания соответствующих индексов показан на рис. 3.

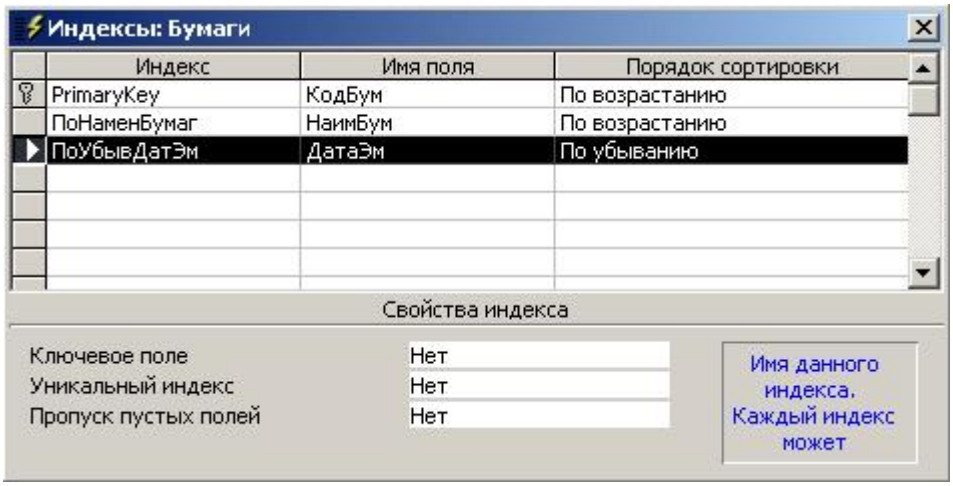

Рис. 3. Создание индексов для таблицы

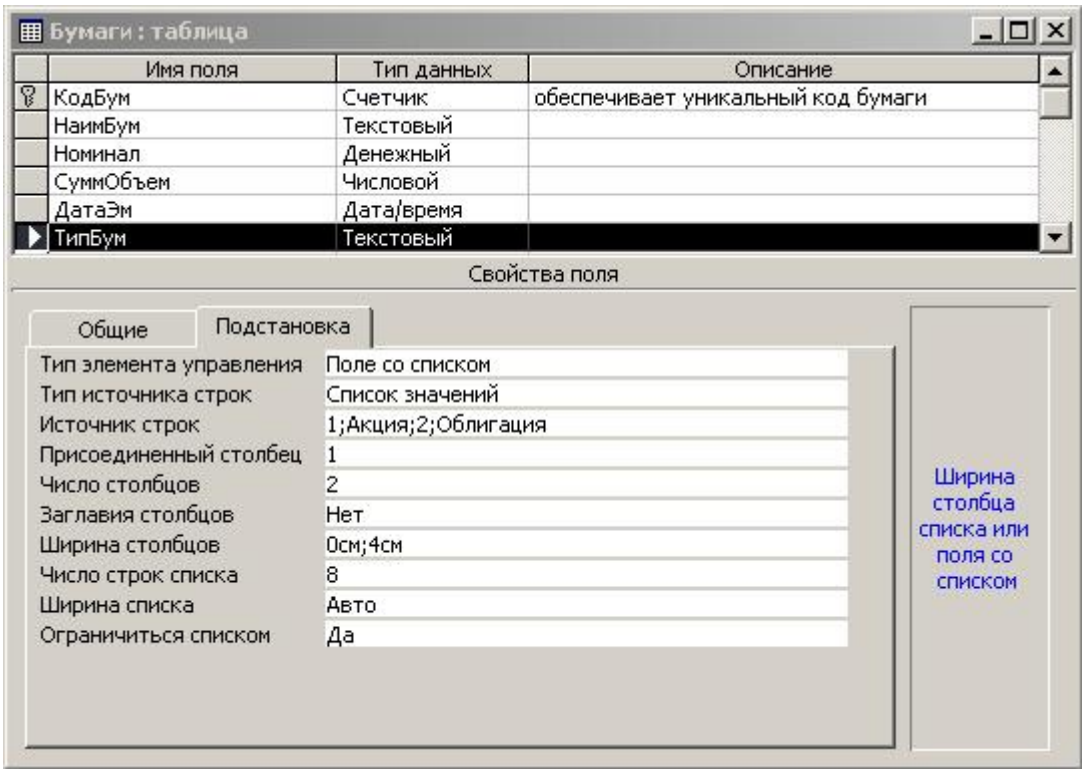

Рис. 4. Задание списка подстановки для поля

Эффективным методом решения задач контроля корректности входных данных является ограничение множества допустимых значений поля некоторым списком. Рассмотрим это более подробно на примере поля ТипБум (тип бумаги), которая, допустим, в рассматриваемой торговой системе может быть либо акцией, либо облигацией. Нетрудно заметить, что будет разумным присвоить типу Акция код 1, а типу Облигация - код 2. Это позволит существенно сэкономить место за счет уменьшения объема хранимой информации (особенно при большом количестве записей). Однако с точки

зрения восприятия вводимой информации пользователем гораздо удобнее иметь дело с осмысленным текстом, чем запоминать, какие коды ему соответствуют.

Средством решения этой проблемы в Access является задание подстановочного списка значений для поля. Для этого следует выбрать вкладку Подстановка в окне Свойства поля, далее для свойства Тип элемента управления задать значение Список. На рис. 4 показано, как описать другие свойства элемента управления Список, чтобы организовать заполнение поля ТипБум требуемыми значениями.

После создания описания структуры таблицы можно перейти в режим непосредственного ввода в нее данных. Как уже говорилось, важным преимуществом интерфейса СУБД Access является продуманная гибкая система перехода от режима Конструктора к режиму ввода данных в таблицу (Режим таблицы). Такой переход можно осуществить, щелкнув мышью по пиктограмме Вид, расположенной на панели инструментов, либо выбрав функцию меню Вид > Режим таблицы.

На рис. 5 показано окно режима непосредственного ввода данных в таблицу Бумаги,

Бумаги : таблица

	КодБум	НаимБум	Номинал	СуммОбъем	Дата эмиссии	ТипБум
	2	ОАО "Автомат"	1 000,00р.	2000	10.01.1993	Акция
▶	3	Красный яр	15 000,00р.	10000	10.01.1993	Облигация
	4	Ока-Банк	0,00р.	0	08.03.1996	Акция
*	(Счетчик)		0,00р.	0	29.05.2002	

Запись: 2 из 3

Рис. 5 Ввод данных в таблицу Бумаги

Хочется еще раз обратить внимание читателя на то, что выбор типа бумаги осуществляется из заранее предопределенного списка.

Создание схемы данных

Очевидно, что те действия, которые были подробно описаны для таблицы Бумаги, следует проделать и для остальных информационных массивов: Агенты, Портфели, Заявки. В результате мы получим систему таблиц базы данных TfadeTest. Подчеркнем, именно систему, так как находящиеся в них данные тесно и содержательно связаны между собой. Действительно, данные, находящиеся в поле Код агента таблицы Портфели, должны быть согласованы по типу и размеру с данными, находящимися в одноименном поле таблицы Бумаги. Более того, логика рассматриваемой задачи требует, чтобы, работая с информацией, относящейся к портфелю, мы могли одновременно обратиться к данным, характеризующим текущего агента, и т. д.

Механизм описания логических связей между таблицами в Access реализован в виде объекта, называемого Схемой данных. Перейти к ее созданию можно из

панели инструментов База данных, доступной из главного окна. Альтернативный вариант вызова данного режима доступен через меню Сервис > Схема данных. Вид, который будет иметь схема данных для построенных на предыдущих шагах таблиц, представлен на рис. 6.

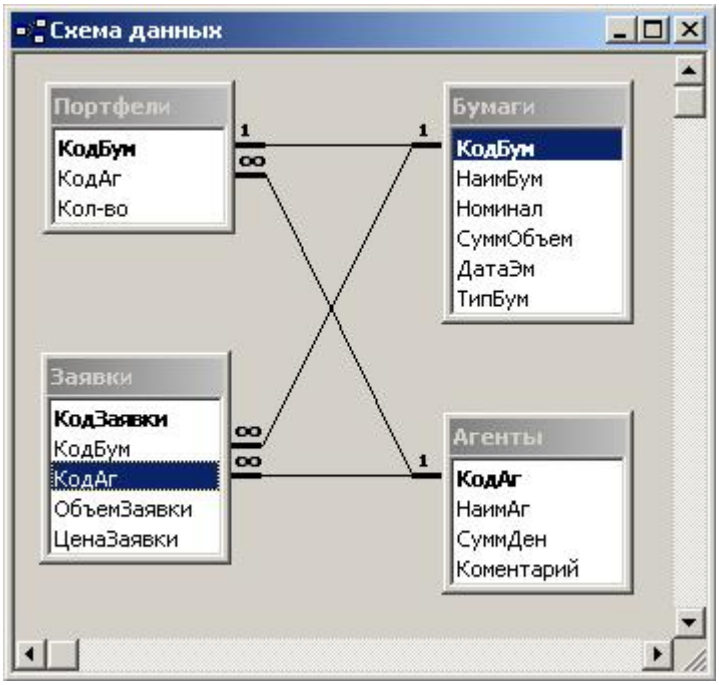

Рис. 6. Создание схемы данных

Интерфейс задания связей между полями в схеме основан на "перетаскивании" (перемещении при нажатой левой кнопки мыши) выбранного поля и "наложении" его на то поле, с которым должна быть установлена связь. Для связывания сразу нескольких полей их следует перемещать при нажатой клавише Ctrl. Выделяют несколько типов связей между таблицами в схеме. " Один к одному" (1:1) - одному значению поля в одной таблице соответствует только одно значение поля в другой. "Один ко многим" (1:?) - одному значению поля в одной таблице соответствует несколько (одно или более) значений в другой.

Важнейшей задачей, которую позволяет решать схема, является обеспечение логической целостности данных в базе. Так, в базе данных TradeTest нарушение целостности может возникнуть в случае удаления из таблицы Бумаги записей по тем бумагам, о которых существуют записи в таблицах Портфели или Заявки, в результате чего в их составе окажутся ссылки на "потерянные" коды. Очевидно, что это можно предотвратить, если каскадно удалить как записи из таблицы Бумаги, так и записи из связанных с ней таблиц. Такой эффект в Access может быть достигнут за счет задания определенных свойств для связи. Чтобы это сделать, необходимо щелкнуть кнопкой мыши, находясь на линии схемы, обозначающей связь. После этого появляется диалоговое окно, предназначенное для изменения свойств связи. Как видно из рис. 7., в рамках режима обеспечения целостности данных можно по выбранной связи задать как

каскадное обновление значений для связанных полей, так и каскадное удаление связанных записей.

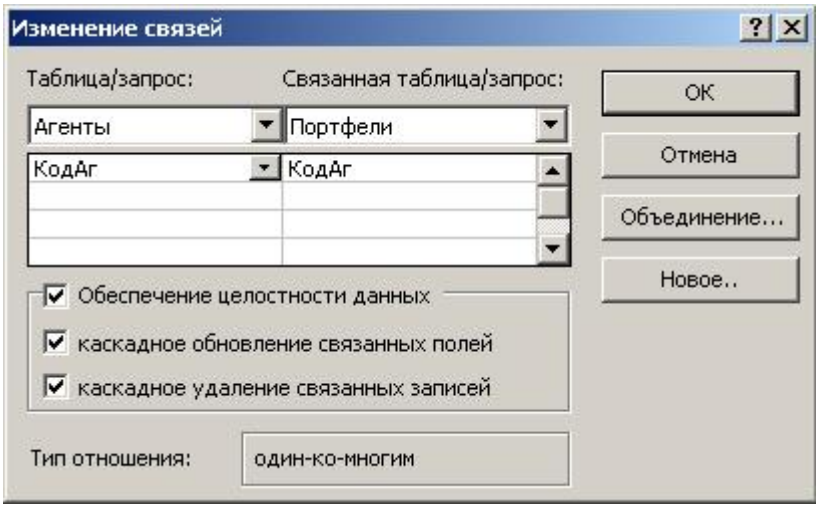

Рис. 7. Задание свойств связи

Вопрос 2. Описание выражений для индексирования ключей, используемых для организации связей между таблицами данных.

Нормализация баз данных

Рассмотрим процесс нормализации базы данных на примере базы данных BIBLIO.MDB. Вообще говоря, все данные о книгах, авторах и издательствах можно разместить в одной таблице, названной, например, BIBLIOS. Структура этой таблицы показана на рис. 1.11. В принципе, можно работать и с такой таблицей. С другой стороны понятно, что такая структура данных является неэффективной. В таблице BIBLIOS будет достаточно много повторяющихся данных, например сведения об издательстве или авторе будут повторяться для каждой опубликованной книги. Такая организация данных приведет к следующим проблемам, с которыми столкнется конечный пользователь вашей программы:

Наличие повторяющихся данных приведет к неоправданному увеличению размера файла базы данных. Кроме нерационального использования дискового пространства, это также вызовет заметное замедление работы приложения.

Ввод пользователем большого количества повторяющейся информации неизбежно приведет к возникновению ошибок.

Изменение, например, номера телефона издательства потребует значительных усилий по корректировке каждой записи, содержащей данные об издателе.

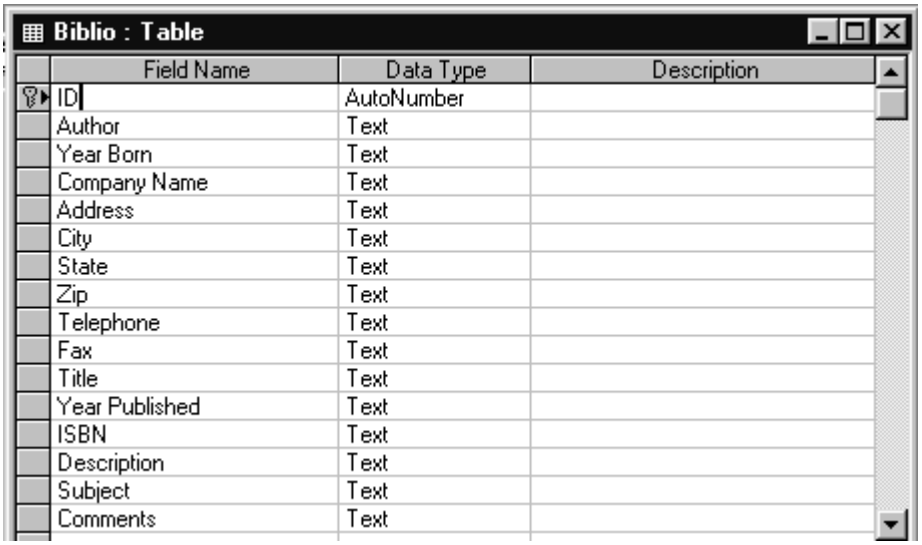

Рис.1.11. Структура таблицы BIBLIOS.

Если, при проектировании приложения для работы с базами данных, вы организуете свои данные таким нерациональным образом, то в дальнейшем вам, скорее всего, больше не поручат решение аналогичных задач.

Чтобы избежать всех этих проблем, надо стремиться максимально уменьшить количество повторяющейся информации. Процесс уменьшения избыточности информации в базе данных посредством разделения ее на несколько связанных друг с другом таблиц и называется нормализацией данных.

Вообще говоря, существует строгая теория нормализации данных, в рамках которой разработаны алгоритмы уменьшения избыточности информации, определены несколько уровней нормализации и установлены критерии, определяющие соответствие данных определенному уровню нормализации. Знакомство с теорией нормализации данных выходит за рамки этих уроков и тем читателям, которым интересно побольше узнать об этом, можно посоветовать обратиться к специальной литературе.

Для того, чтобы построить достаточно эффективную структуру данных, достаточно придерживаться нескольких простых правил:

1. Определите таблицы таким образом, чтобы записи в каждой таблице описывали объекты одного и того же типа. В нашем случае библиографические данные можно разместить в трех таблицах:

- PUBLISHERS - содержит информацию об издательствах;
- AUTHORS - содержит информацию об авторах книг;
- TITLES - содержит информацию об изданных книгах.

2. Если в вашей таблице появляются поля, содержащие аналогичные данные, разделите таблицу.

3. Не запоминайте в таблице данных, которые могут быть вычислены при помощи данных из других таблиц.

4. Используйте вспомогательные таблицы. Например, если в вашей таблице есть поле Страна, то может быть стоит ввести вспомогательную таблицу Country, которая будет содержать соответствующие записи (Россия, Украина, США и т.п.). Этот прием также поможет уменьшить количество ошибок при вводе данных, допускаемых пользователями.

Ключи и индексы

Как было отмечено выше при описании отношений между таблицами, в реляционных базах данных таблицы связываются друг с другом посредством совпадающих значений ключевых полей. Ключевым полем может быть практически любое поле в таблице. Ключ может быть первичным (primary) или внешним (foreign).

Первичный ключ однозначно определяет запись в таблице. В примере с базой данных BIBLIO.MDB таблицы PUBLISHERS, AUTHORS и TITLES имеют первичные ключи PubID, Au_ID и ISBN соответственно. Таблица TITLES также имеет два внешних ключа PubID и Au_ID для связи с таблицами PUBLISHERS и AUTHORS. Таким образом, первичный ключ однозначно определяет запись в таблице, в то время как внешний ключ используется для связи с первичным ключом другой таблицы.

Ключевой поле может иметь определенный смысл, как например ключ ISBN в таблице TITLES. Однако, очень часто ключевое поле не несет никакой смысловой нагрузки и является просто идентификатором объекта в таблице. Во многих случаях удобно использовать в качестве ключа поле счетчика (Counter field). При этом вся ответственность по поддержанию уникальности ключевого поля снимается с пользователя и перекладывается на процессор баз данных. Поле счетчика представляет собой четырехбайтовое целое число (Long) и автоматически увеличивается на единицу при добавлении пользователем новой записи в таблицу.

Данные запоминаются в таблице в том порядке, в котором они вводятся пользователем. Это, так называемый, физический порядок следования записей. Однако, часто требуется представить данные в другом, отличном от физического, порядке. Например может потребоваться просмотреть данные об авторах книг, упорядоченные по алфавиту. Кроме того, часто необходимо найти в большом объеме информации запись, удовлетворяющую определенному критерию. Простой перебор записей при поиске в большой таблице может потребовать достаточно много времени и поэтому будет неэффективным.

Одними из основных требований, предъявляемым к системам управления базами данных, являются возможность представления данных в определенном, отличном от физического, порядке и возможность быстрого поиска определенной записи. Эффективным средством решения этих задач является использование индексов.

Индекс представляет собой таблицу, которая содержит ключевые значения для каждой записи в таблице данных и записанные в порядке, требуемом для пользователя. Ключевые значения определяются на основе одного или нескольких полей таблицы. Кроме того, индекс содержит уникальные ссылки на соответствующие записи в таблице. На рис.1.12 показан фрагмент таблицы CUSTOMERS, содержащей информацию о покупателях, и индекс IDX_NAME, построенный на основе поля Name таблицы CUSTOMERS. Индекс IDX_NAME содержит значения ключевого поля Name, упорядоченные в алфавитном порядке, и ссылки на соответствующие записи в таблице CUSTOMERS.

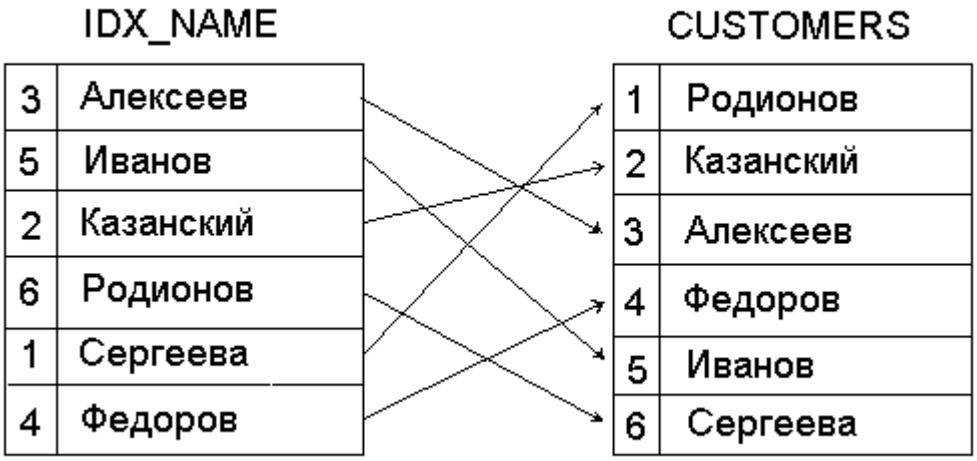

Рис.1.12. Связь между таблицей и индексом.

Каждая таблица может иметь несколько различных индексов, каждый из которых определяет свой собственный порядок следования записей. Например, таблица AUTHORS может иметь индексы для представления данных об авторах, упорядоченные по дате рождения или по алфавиту. Таким образом, каждый индекс используется для представления одних и тех же данных, но упорядоченных различным образом.

Вообще говоря, таблицы в базе данных могут и не иметь индексов. В этом случае для большой таблицы время поиска определенной записи может быть весьма значительным и использование индекса становиться необходимым. С другой стороны, не следует увлекаться созданием слишком большого количества индексов, так как это может заметно увеличить время необходимое

для обновления базы данных и значительно увеличить размер файла базы данных.

При разработке приложений, работающих с базами данных, наиболее широко используются простые индексы. Простые индексы используют значения одного поля таблицы. Примером простого индекса в базе данных BIBLIO.MDB может служить код ISBN, идентификатор автора Au_ID или идентификатор издательства PubID.

Хотя в большинстве случаев для представления данных в определенном порядке достаточно использовать простой индекс, часто возникают ситуации, где не обойтись без использования составных индексов. Составной индекс строится на основе значений двух или более полей таблицы. Хорошей иллюстрацией использования составных индексов может служить база данных родственников при генеалогических исследованиях какой-либо фамилии. Понятно, что использование в качестве простого индекса фамилии человека в данном случае недопустимо. Даже использование составного индекса, основанного на полях имени, фамилии и отчества может быть неэффективным, так как и в этом случае все равно возможно существование достаточно большого числа однофамильцев. Выходом из положения может быть использование составного индекса, основанного, например, на следующих полях таблицы: имя, фамилия, отчество, город и номер телефона.

III. Заключительная часть

При подведении итогов преподаватель анализирует степень реализации поставленных целей занятия, выставляет слушателям оценки за выполнение практических заданий и ответы на контрольные вопросы, выдаёт задания на самостоятельную подготовку.

Контрольные вопросы:

1. Понятие, типы и элементы выражений.
2. Функции преобразований типов данных.
3. Типы индексов и порядок их использования при организации связей между таблицами данными.
4. Понятие триггера и его использование при контроле данных.

Литература

1. Попов В.Б. Основы информационных и телекоммуникационных технологий.- Финансы и статистика, 2007 г. – 336 с.
2. Угринович Н.Д. Информационные технологии и компьютеризация делопроизводства. – М.:БИНОМ ЛЗ.,2007.-394с.

3. Шамраев А.В. Правовое регулирование информационных технологий (анализ проблем и основные документы). – Статут, 2003. – 1013 с.

4. Хроленко А.Т., Денисов А.В. Современные информационные технологии для гуманитария. – Флинта, 2007. – 128 с.

5. Федотова Е.Л. Информационные технологии в профессиональной деятельности. – Форум, 2008. – 368 с.

6. Емельянова Н.З., Партыка Т.Л., Попов И.И. Основы построения автоматизированных информационных систем. – Форум, 2007. – 416 с.

7. Беляева Т. М., Важнов С. А., Вешняков В. В., Кудинов А. Т., Пальянова Н. В. Основы информатики и математики для юристов. – Элит, 2007. – 368 с.

8. Казанцев С.Я. Информатика и математика для юристов. Учебник. – Юнити, 2008. – 560 с.

9. Строганов М.П., Щербаков М.А. Информационные сети и телекоммуникации. – Высшая школа, 2008. – 151 с.

10. Микрюков В.Ю. Информация, информатика, компьютер, информационные системы, сети. – Феникс, 2007. – 448 с.